For answer script - **EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.**

Name:	Tsoi Yiu Chik	Student No.:	20195601A
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**1**	Total no. of pages:	**18**

Declaration of Original Work

By submitting the answer script of this assignment to the subject lecturer through Moodle Centralized Group, you hereby declare that the work in the answer sheet is completely your own work. No part of the answer sheet is taken from other people's work without giving them credit. All references have been clearly cited.

You understand that an infringement of this declaration leaves you subject to disciplinary actions such as mark deduction, disqualification or even expulsion by the College.

If necessary, students may be invited to provide more information on their submission.

(Please refer to the relevant section(s) on plagiarism of the Student Handbook.)

Instructions to Students:

1. Please refer to assignment specification for the submission method
2. Show all your work clearly and neatly. Marks will be deducted for untidy work.

Answer ALL questions.

Answer for Question 1

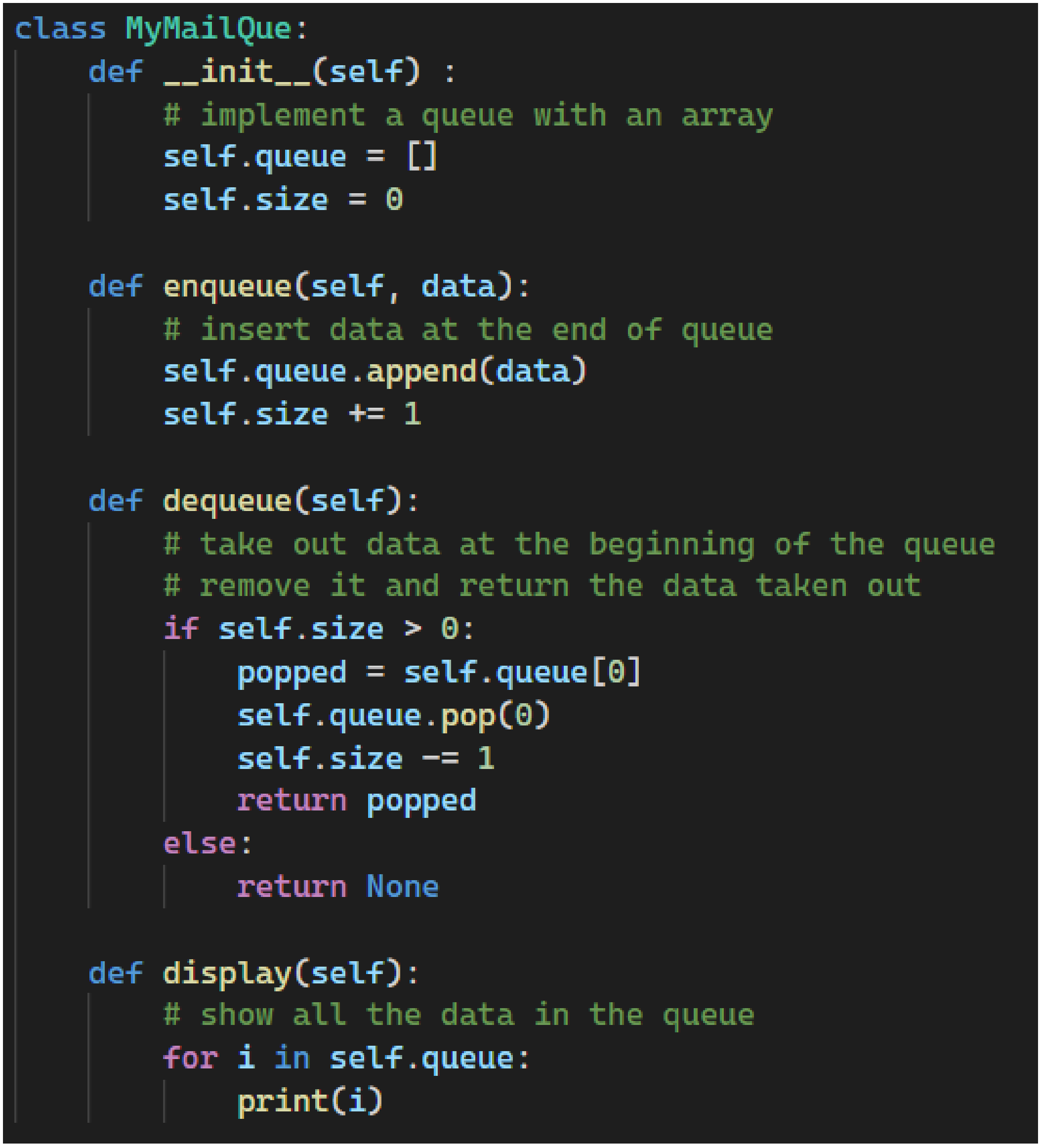
A) + * a b * c d ➔ + * 95 56 * 60 1
B) a * b + c * d ➔ 95 * 56 + 60 * 1
C) 5380

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601A
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**3**	Total no. of pages:	**18**

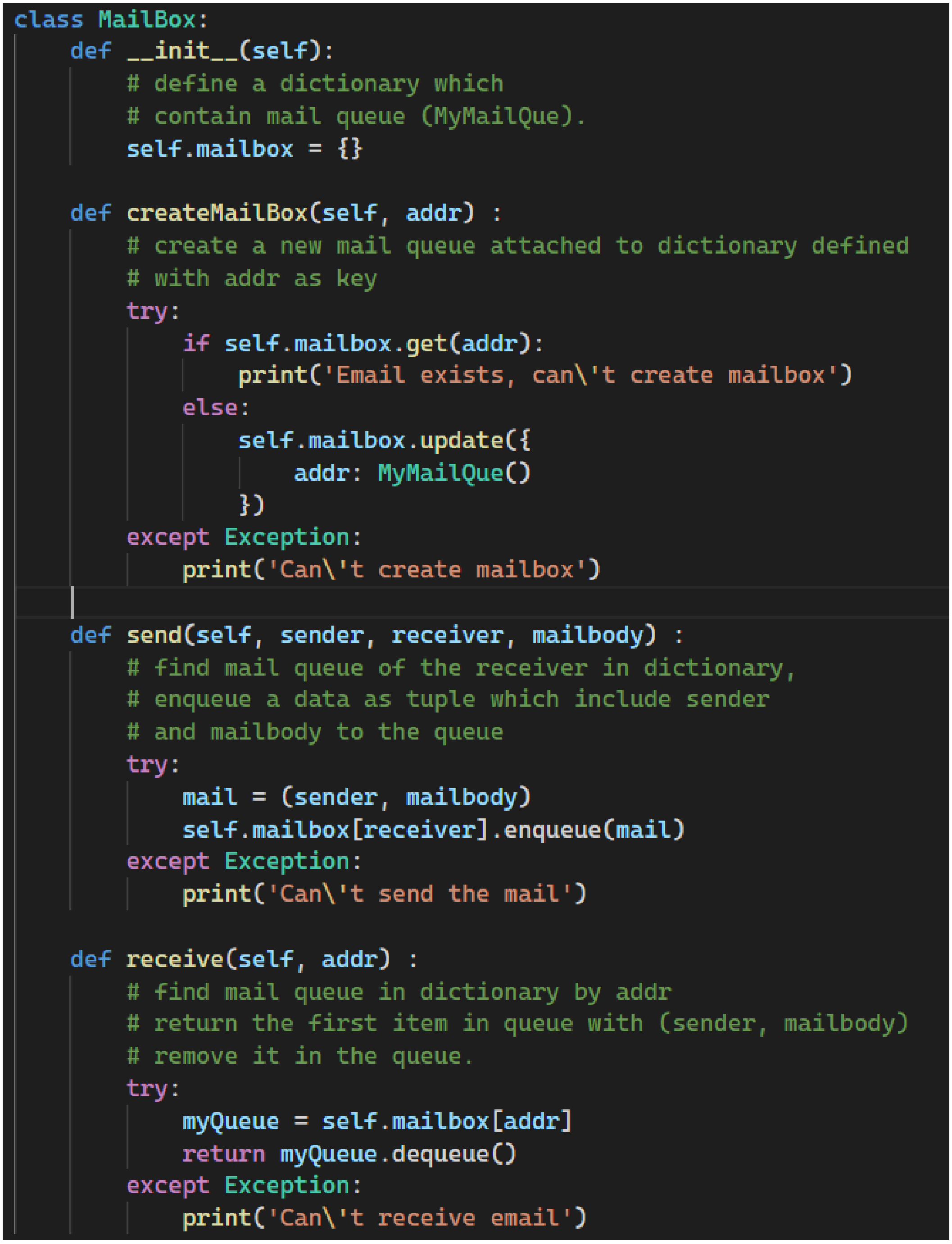
Answer for Question 2

A)



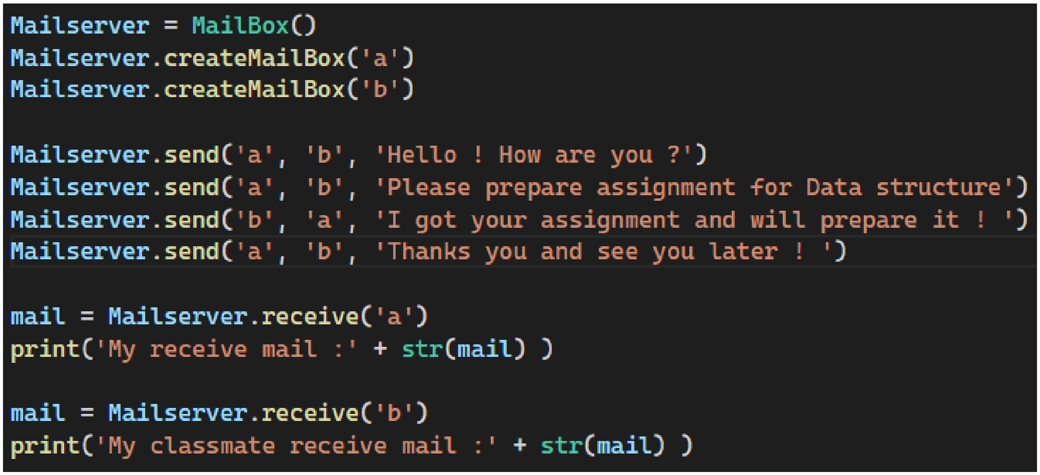
```
class MyMailQue:
    def __init__(self) :
        # implement a queue with an array
        self.queue = []
        self.size = 0

    def enqueue(self, data):
        # insert data at the end of queue
        self.queue.append(data)
        self.size += 1

    def dequeue(self):
        # take out data at the beginning of the queue
        # remove it and return the data taken out
        if self.size > 0:
            popped = self.queue[0]
            self.queue.pop(0)
            self.size -= 1
            return popped
        else:
            return None

    def display(self):
        # show all the data in the queue
        for i in self.queue:
            print(i)
```

B)

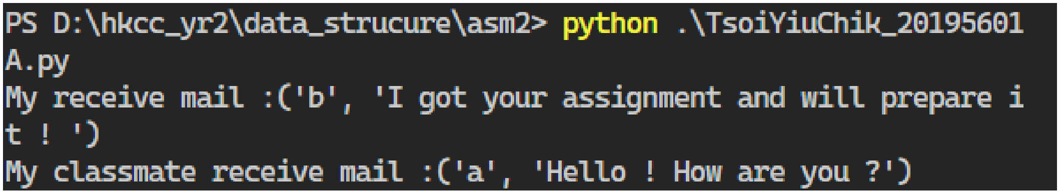


```
class MailBox:
    def __init__(self):
        # define a dictionary which
        # contain mail queue (MyMailQue).
        self.mailbox = {}

    def createMailBox(self, addr) :
        # create a new mail queue attached to dictionary defined
        # with addr as key
        try:
            if self.mailbox.get(addr):
                print('Email exists, can\'t create mailbox')
            else:
                self.mailbox.update({
                    addr: MyMailQue()
                })
        except Exception:
            print('Can\'t create mailbox')

    def send(self, sender, receiver, mailbody) :
        # find mail queue of the receiver in dictionary,
        # enqueue a data as tuple which include sender
        # and mailbody to the queue
        try:
            mail = (sender, mailbody)
            self.mailbox[receiver].enqueue(mail)
        except Exception:
            print('Can\'t send the mail')

    def receive(self, addr) :
        # find mail queue in dictionary by addr
        # return the first item in queue with (sender, mailbody)
        # remove it in the queue.
        try:
            myQueue = self.mailbox[addr]
            return myQueue.dequeue()
        except Exception:
            print('Can\'t receive email')
```

C)

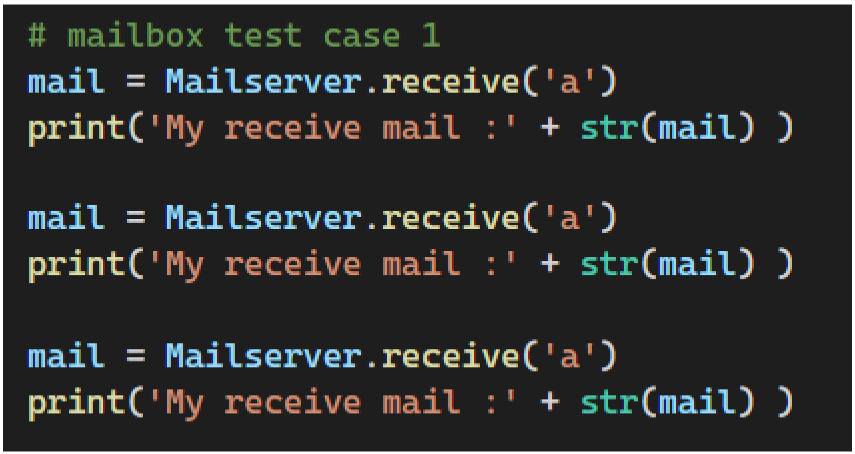


```
Mailserver = MailBox()
Mailserver.createMailBox('a')
Mailserver.createMailBox('b')

Mailserver.send('a', 'b', 'Hello ! How are you ?')
Mailserver.send('a', 'b', 'Please prepare assignment for Data structure')
Mailserver.send('b', 'a', 'I got your assignment and will prepare it ! ')
Mailserver.send('a', 'b', 'Thanks you and see you later ! ')

mail = Mailserver.receive('a')
print('My receive mail :' + str(mail) )

mail = Mailserver.receive('b')
print('My classmate receive mail :' + str(mail) )
```

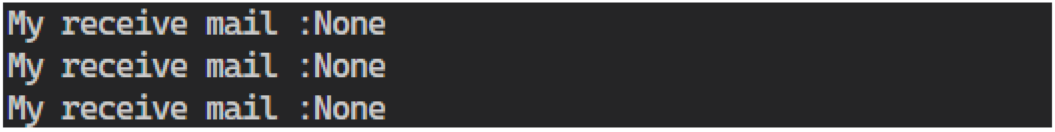


```
PS D:\hkcc_yr2\data_strucure\asm2> python .\TsoiYiuChik_20195601
A.py
My receive mail :('b', 'I got your assignment and will prepare i
t ! ')
My classmate receive mail :('a', 'Hello ! How are you ?')
```

Test case 1:

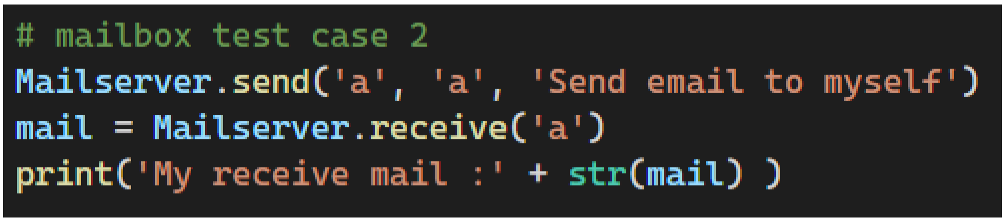


```
# mailbox test case 1
mail = Mailserver.receive('a')
print('My receive mail :' + str(mail) )

mail = Mailserver.receive('a')
print('My receive mail :' + str(mail) )

mail = Mailserver.receive('a')
print('My receive mail :' + str(mail) )
```

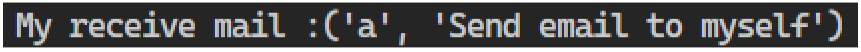


```
My receive mail :None
My receive mail :None
My receive mail :None
```

Test case 2:

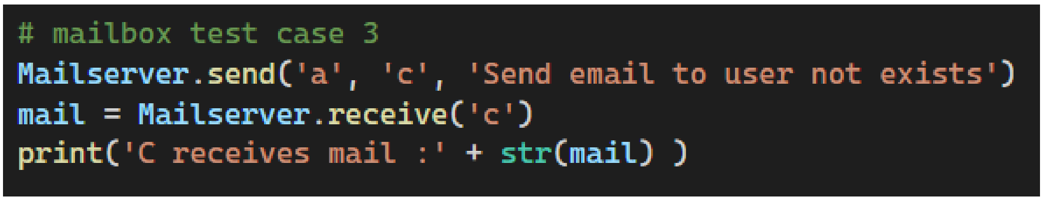


```
# mailbox test case 2
Mailserver.send('a', 'a', 'Send email to myself')
mail = Mailserver.receive('a')
print('My receive mail :' + str(mail) )
```

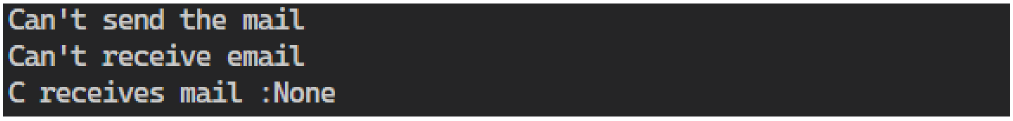


```
My receive mail :('a', 'Send email to myself')
```

Test case 3:



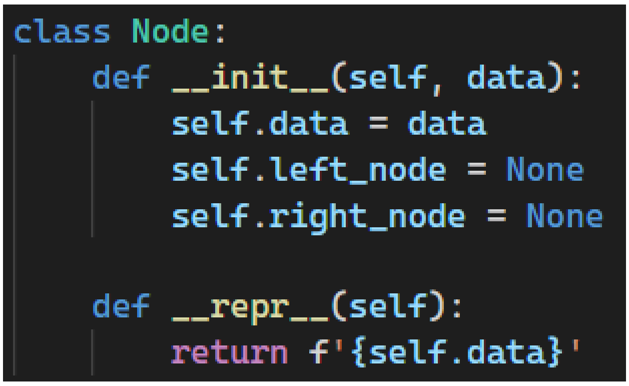
```
# mailbox test case 3
Mailserver.send('a', 'c', 'Send email to user not exists')
mail = Mailserver.receive('c')
print('C receives mail :' + str(mail) )
```



```
Can't send the mail
Can't receive email
C receives mail :None
```

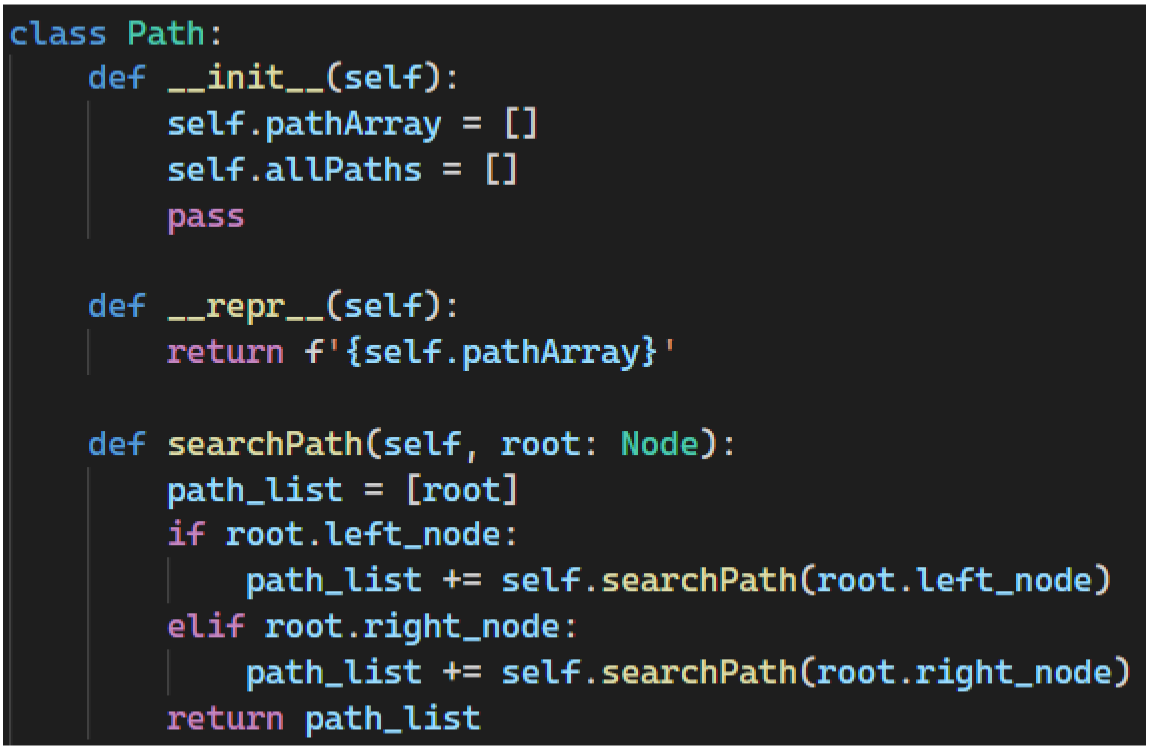
Answer for Question 3

A)i)



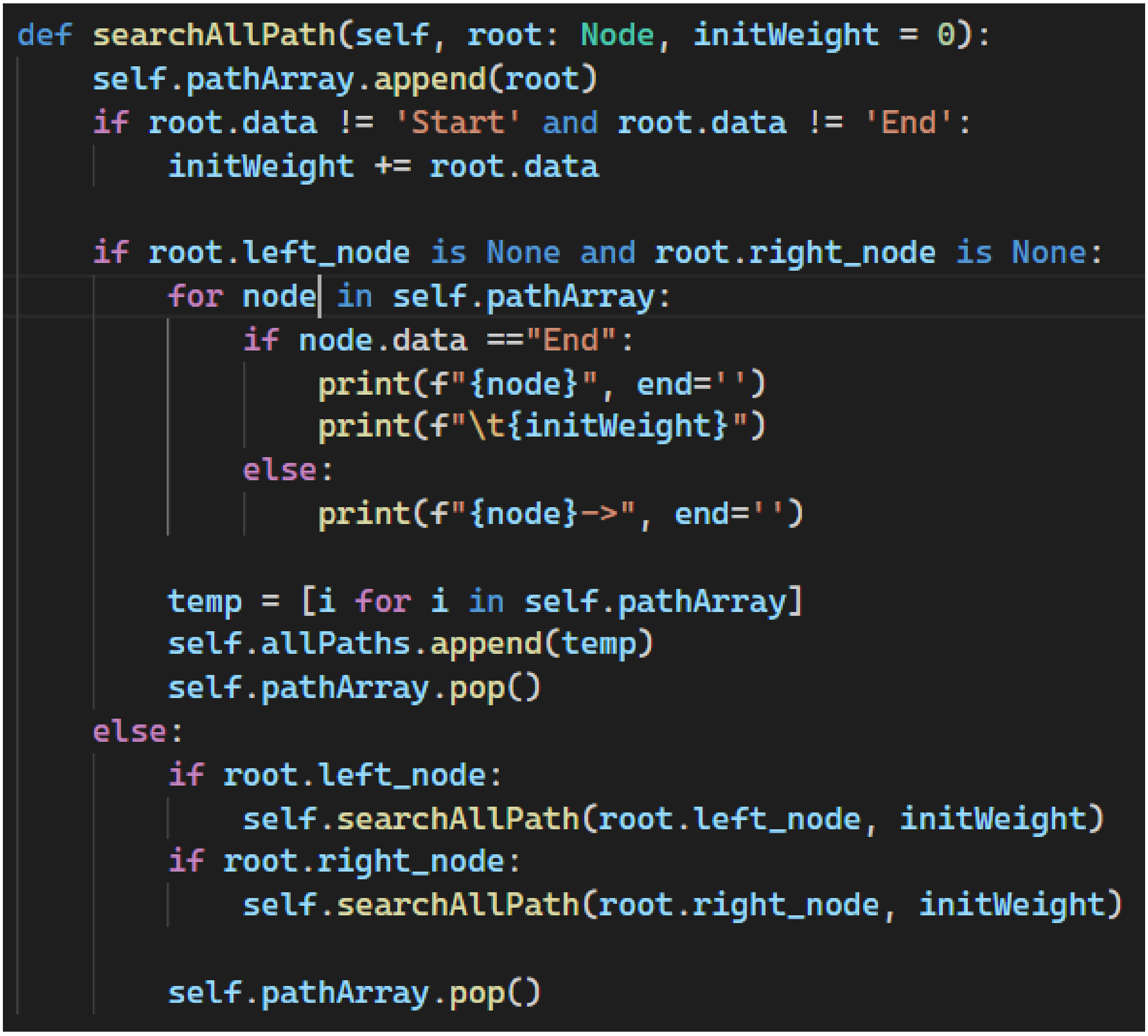
```python
class Node:
    def __init__(self, data):
        self.data = data
        self.left_node = None
        self.right_node = None

    def __repr__(self):
        return f'{self.data}'
```

ii)



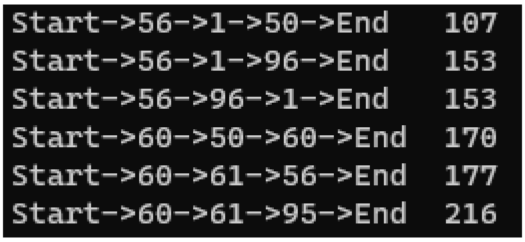
```python
class Path:
    def __init__(self):
        self.pathArray = []
        self.allPaths = []
        pass

    def __repr__(self):
        return f'{self.pathArray}'

    def searchPath(self, root: Node):
        path_list = [root]
        if root.left_node:
            path_list += self.searchPath(root.left_node)
        elif root.right_node:
            path_list += self.searchPath(root.right_node)
        return path_list
```

B)



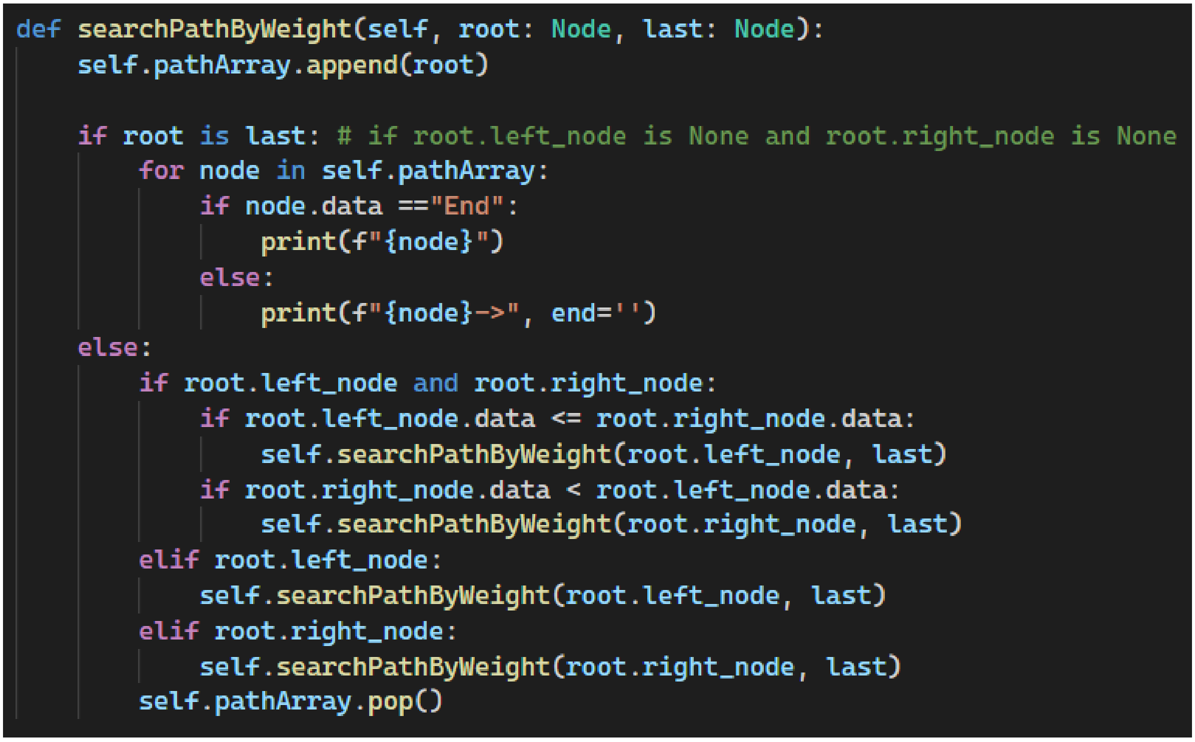
```python
def searchAllPath(self, root: Node, initWeight = 0):
    self.pathArray.append(root)
    if root.data != 'Start' and root.data != 'End':
        initWeight += root.data

    if root.left_node is None and root.right_node is None:
        for node in self.pathArray:
            if node.data =="End":
                print(f"{node}", end='')
                print(f"\t{initWeight}")
            else:
                print(f"{node}->", end='')

        temp = [i for i in self.pathArray]
        self.allPaths.append(temp)
        self.pathArray.pop()
    else:
        if root.left_node:
            self.searchAllPath(root.left_node, initWeight)
        if root.right_node:
            self.searchAllPath(root.right_node, initWeight)

        self.pathArray.pop()
```

searchAllPath() output:



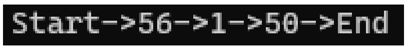
```
Start->56->1->50->End    107
Start->56->1->96->End    153
Start->56->96->1->End    153
Start->60->50->60->End   170
Start->60->61->56->End   177
Start->60->61->95->End   216
```

C)



```
def searchPathByWeight(self, root: Node, last: Node):
    self.pathArray.append(root)

    if root is last: # if root.left_node is None and root.right_node is None
        for node in self.pathArray:
            if node.data =="End":
                print(f"{node}")
            else:
                print(f"{node}->", end='')
    else:
        if root.left_node and root.right_node:
            if root.left_node.data <= root.right_node.data:
                self.searchPathByWeight(root.left_node, last)
            if root.right_node.data < root.left_node.data:
                self.searchPathByWeight(root.right_node, last)
        elif root.left_node:
            self.searchPathByWeight(root.left_node, last)
        elif root.right_node:
            self.searchPathByWeight(root.right_node, last)
        self.pathArray.pop()
```

searchPathByWeight() output:



```
Start->56->1->50->End
```

D)

searchAllPath() disadvantage:

As the searchAllPath() method walks through all the nodes of the graph, time complexity is higher(i.e. O(n)), efficiency decreases when the number of nodes grows.

searchAllPath() advantage:

As all nodes have been traversed, weight of every path has been listed out, the dedicated path with least weight can be chosen easily.

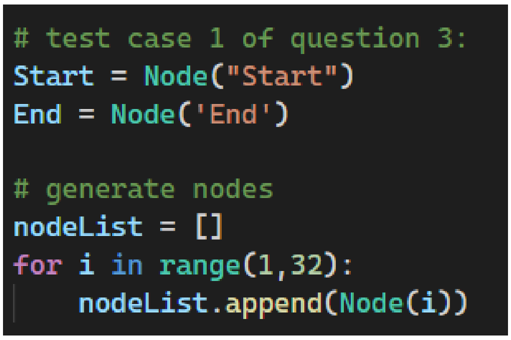
searchPathByWeight() advantage:

In this method, only one path has been traversed, the time complexity will only remain O(log n) in this structure

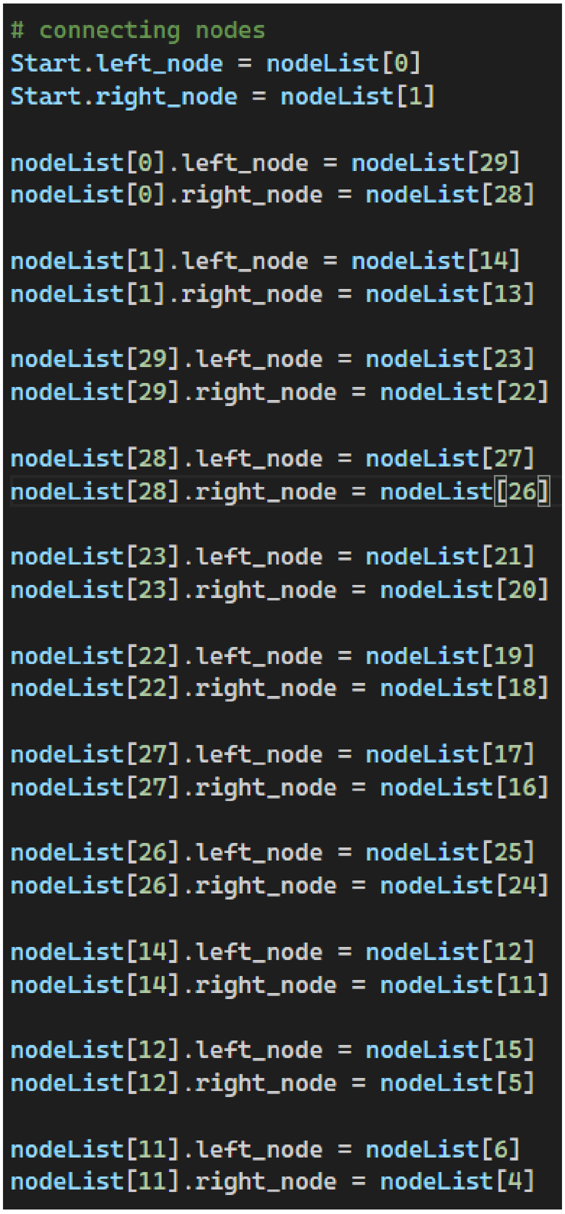
searchPathByWeight() disadvantage:

Since the graph is not a BST, it is not guaranteed that the chosen path must be the one with least weight. Test case 1 demonstrates this argument.

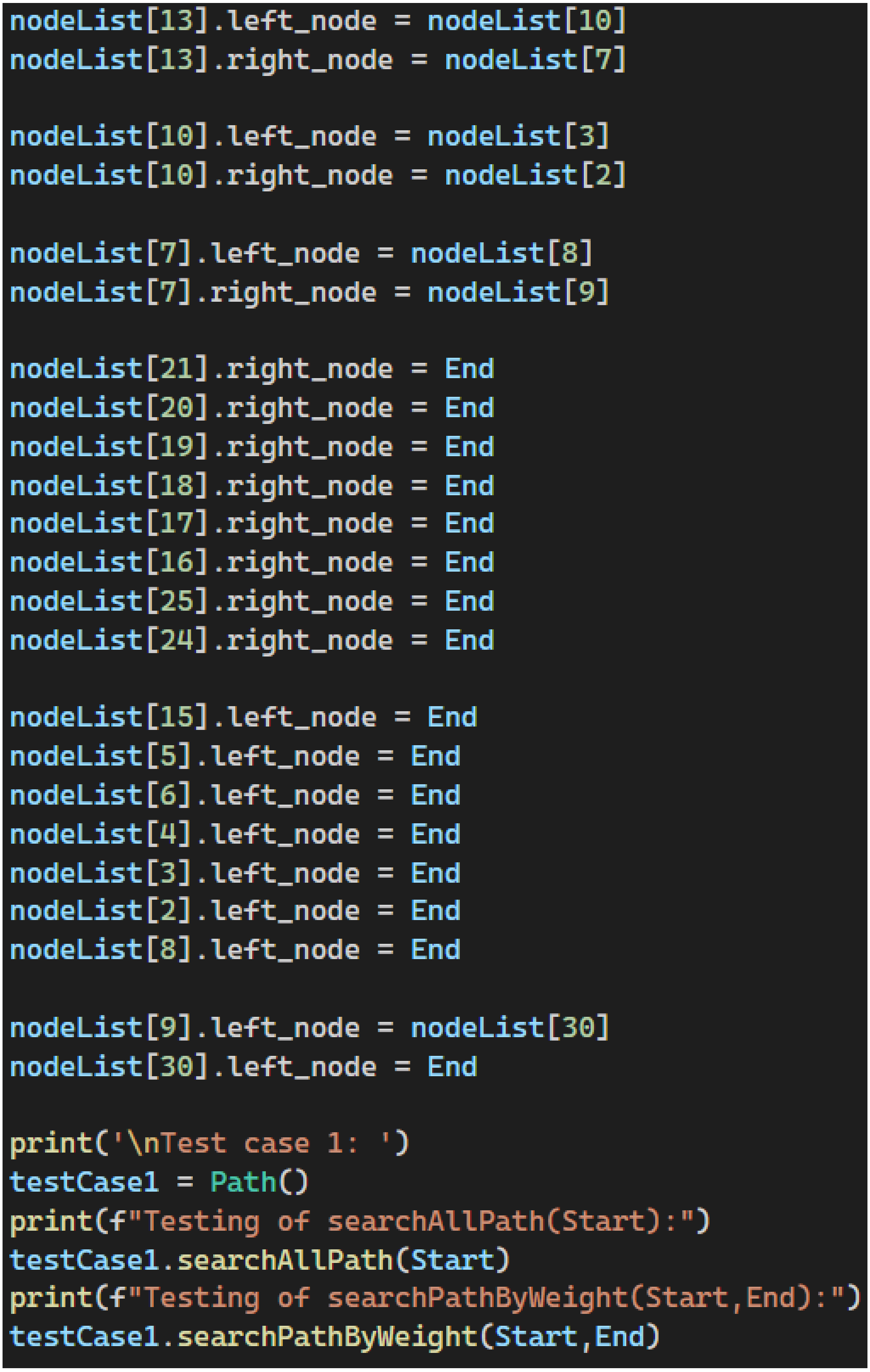
Test case 1 of question 3:



```
# test case 1 of question 3:
Start = Node("Start")
End = Node('End')

# generate nodes
nodeList = []
for i in range(1,32):
    nodeList.append(Node(i))
```

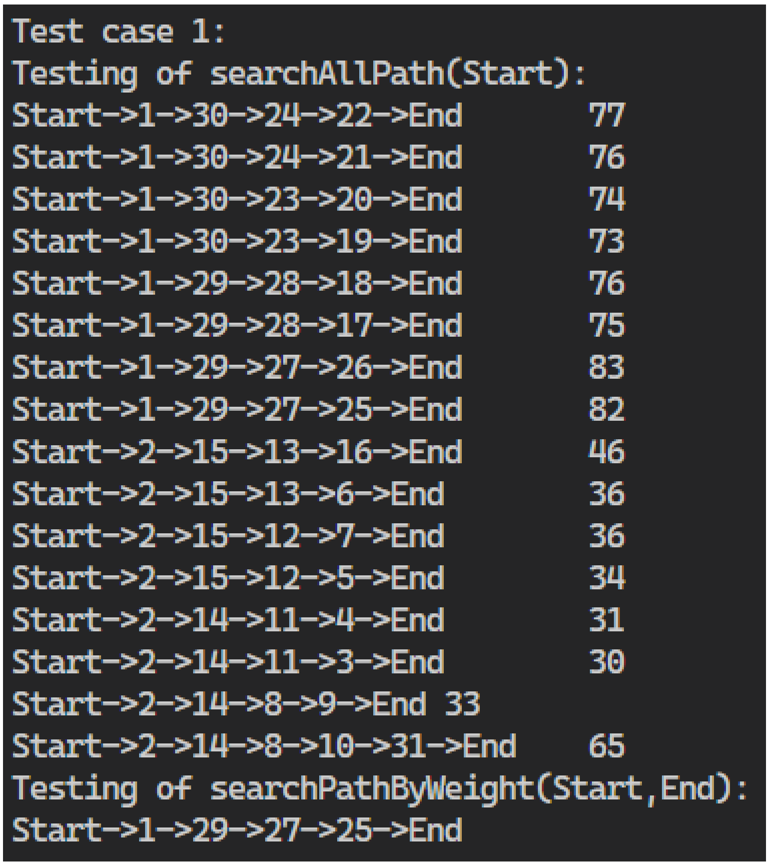


```
# connecting nodes
Start.left_node = nodeList[0]
Start.right_node = nodeList[1]

nodeList[0].left_node = nodeList[29]
nodeList[0].right_node = nodeList[28]

nodeList[1].left_node = nodeList[14]
nodeList[1].right_node = nodeList[13]

nodeList[29].left_node = nodeList[23]
nodeList[29].right_node = nodeList[22]

nodeList[28].left_node = nodeList[27]
nodeList[28].right_node = nodeList[26]

nodeList[23].left_node = nodeList[21]
nodeList[23].right_node = nodeList[20]

nodeList[22].left_node = nodeList[19]
nodeList[22].right_node = nodeList[18]

nodeList[27].left_node = nodeList[17]
nodeList[27].right_node = nodeList[16]

nodeList[26].left_node = nodeList[25]
nodeList[26].right_node = nodeList[24]

nodeList[14].left_node = nodeList[12]
nodeList[14].right_node = nodeList[11]

nodeList[12].left_node = nodeList[15]
nodeList[12].right_node = nodeList[5]

nodeList[11].left_node = nodeList[6]
nodeList[11].right_node = nodeList[4]
```



```
nodeList[13].left_node = nodeList[10]
nodeList[13].right_node = nodeList[7]

nodeList[10].left_node = nodeList[3]
nodeList[10].right_node = nodeList[2]

nodeList[7].left_node = nodeList[8]
nodeList[7].right_node = nodeList[9]

nodeList[21].right_node = End
nodeList[20].right_node = End
nodeList[19].right_node = End
nodeList[18].right_node = End
nodeList[17].right_node = End
nodeList[16].right_node = End
nodeList[25].right_node = End
nodeList[24].right_node = End

nodeList[15].left_node = End
nodeList[5].left_node = End
nodeList[6].left_node = End
nodeList[4].left_node = End
nodeList[3].left_node = End
nodeList[2].left_node = End
nodeList[8].left_node = End

nodeList[9].left_node = nodeList[30]
nodeList[30].left_node = End

print('\nTest case 1: ')
testCase1 = Path()
print(f"Testing of searchAllPath(Start):")
testCase1.searchAllPath(Start)
print(f"Testing of searchPathByWeight(Start,End):")
testCase1.searchPathByWeight(Start,End)
```

Test case 1 output:

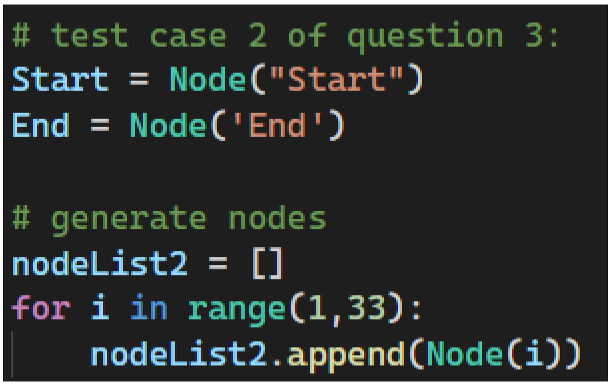


```
Test case 1:
Testing of searchAllPath(Start):
Start->1->30->24->22->End       77
Start->1->30->24->21->End       76
Start->1->30->23->20->End       74
Start->1->30->23->19->End       73
Start->1->29->28->18->End       76
Start->1->29->28->17->End       75
Start->1->29->27->26->End       83
Start->1->29->27->25->End       82
Start->2->15->13->16->End       46
Start->2->15->13->6->End        36
Start->2->15->12->7->End        36
Start->2->15->12->5->End        34
Start->2->14->11->4->End        31
Start->2->14->11->3->End        30
Start->2->14->8->9->End 33
Start->2->14->8->10->31->End    65
Testing of searchPathByWeight(Start,End):
Start->1->29->27->25->End
```

Path weight of searchPathByWeight: 82

Path with least weight in searchAllPath: 30

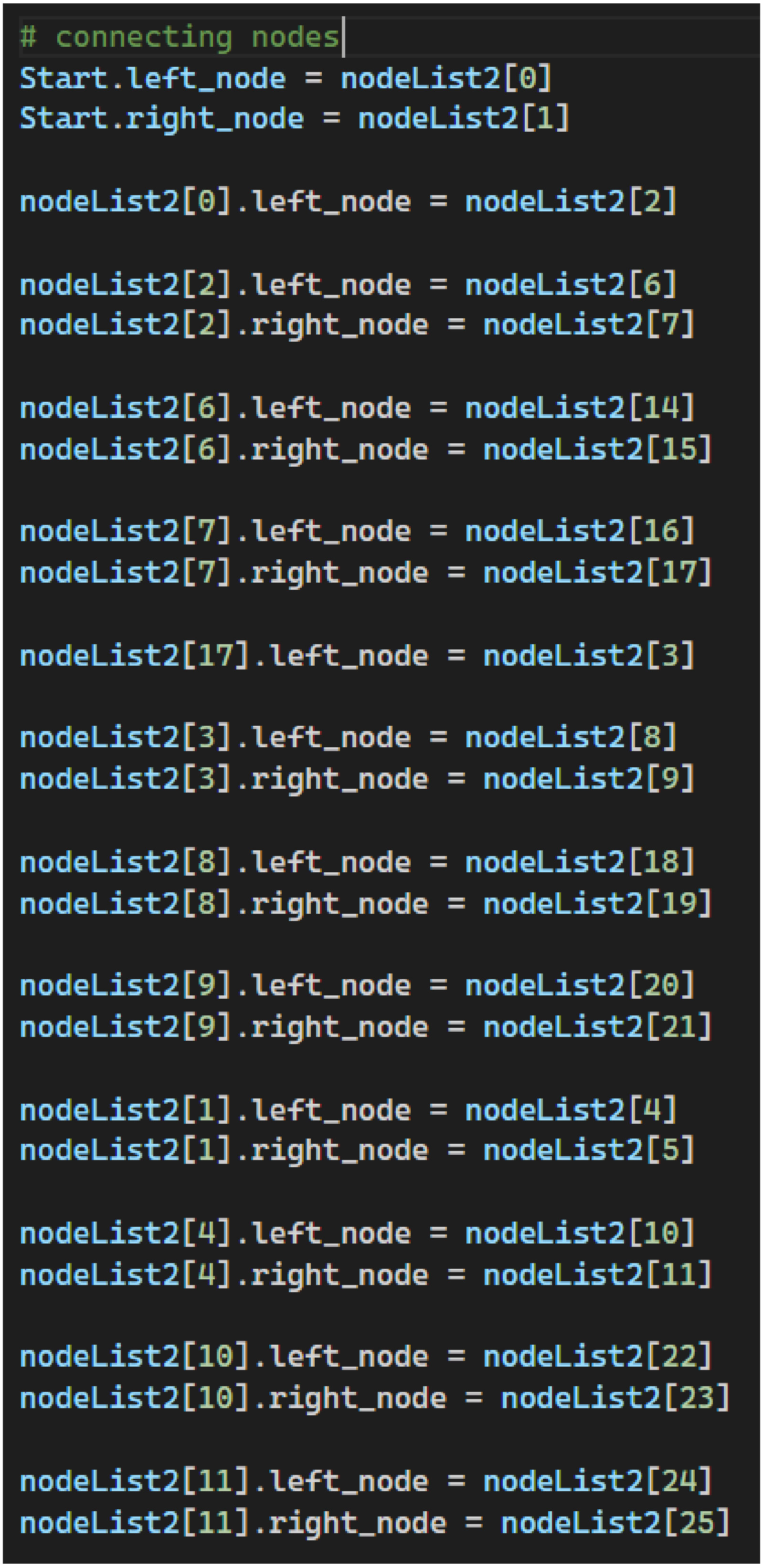
Test case 2:



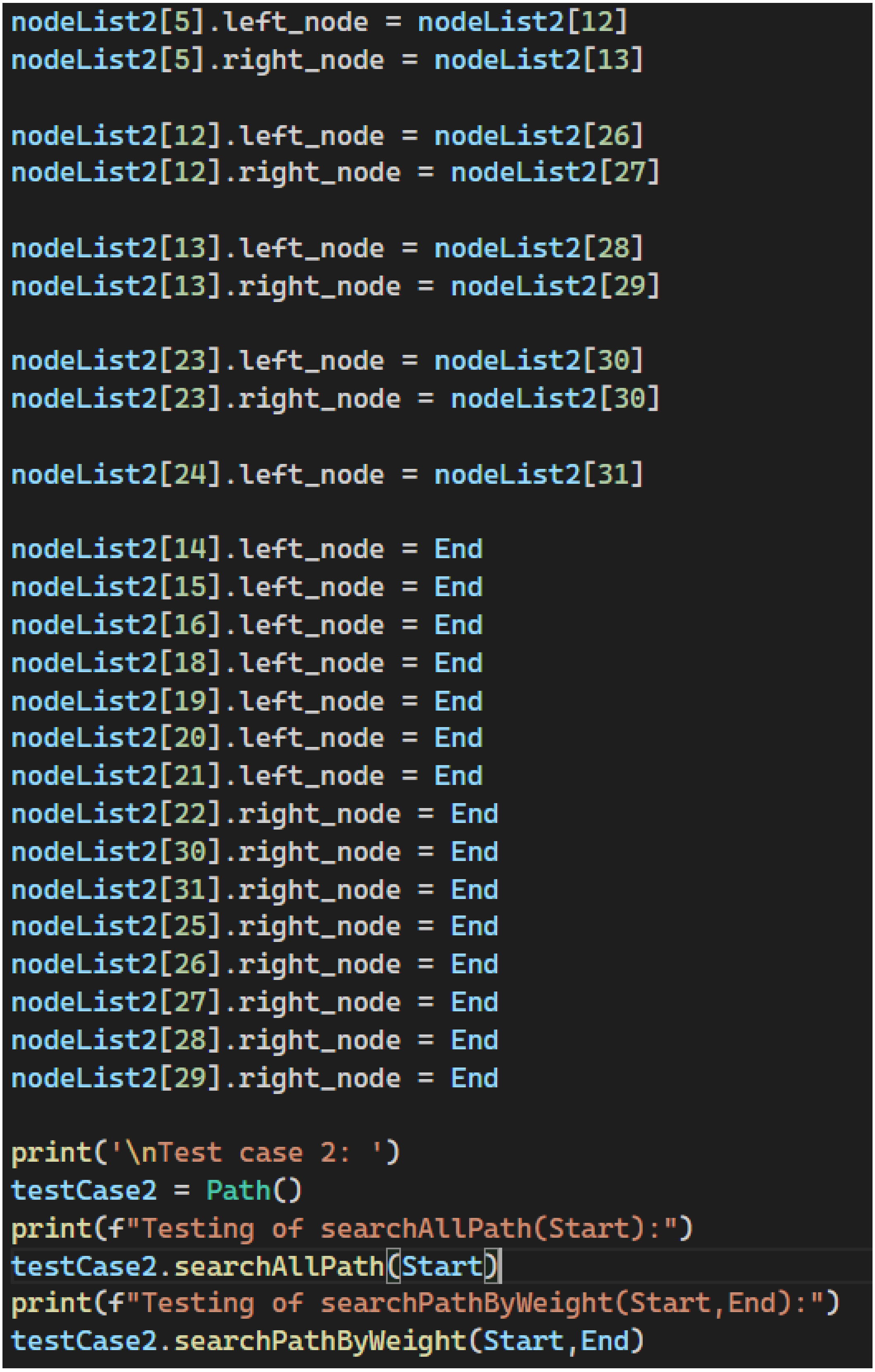
```
# test case 2 of question 3:
Start = Node("Start")
End = Node('End')

# generate nodes
nodeList2 = []
for i in range(1,33):
    nodeList2.append(Node(i))
```

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601A
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**14**	Total no. of pages:	**18**

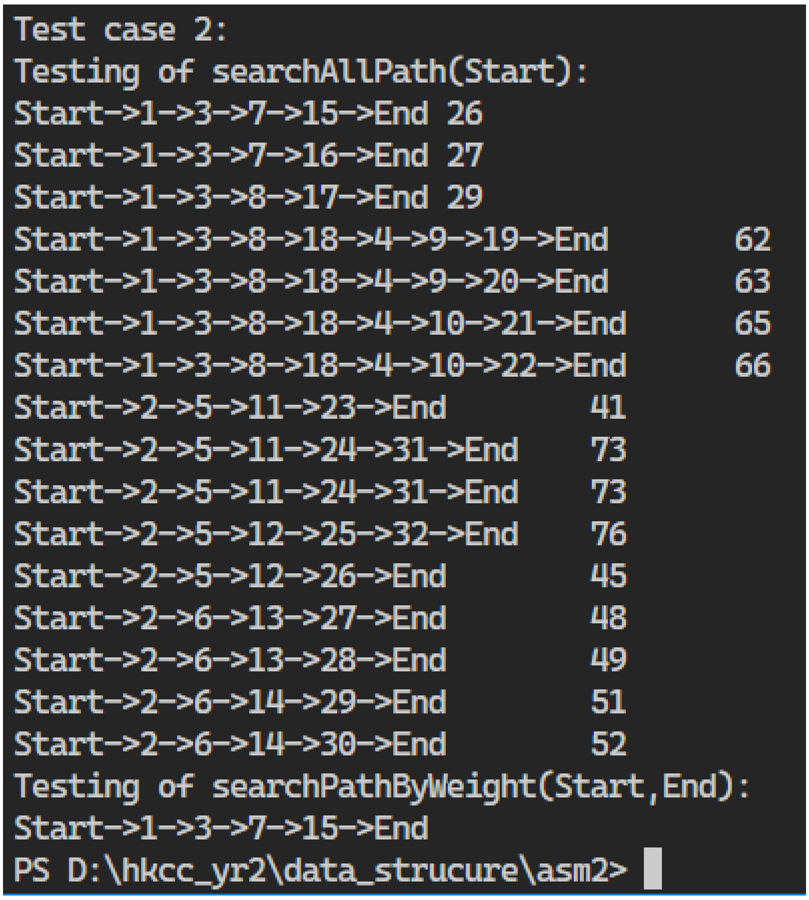


```
# connecting nodes
Start.left_node = nodeList2[0]
Start.right_node = nodeList2[1]

nodeList2[0].left_node = nodeList2[2]

nodeList2[2].left_node = nodeList2[6]
nodeList2[2].right_node = nodeList2[7]

nodeList2[6].left_node = nodeList2[14]
nodeList2[6].right_node = nodeList2[15]

nodeList2[7].left_node = nodeList2[16]
nodeList2[7].right_node = nodeList2[17]

nodeList2[17].left_node = nodeList2[3]

nodeList2[3].left_node = nodeList2[8]
nodeList2[3].right_node = nodeList2[9]

nodeList2[8].left_node = nodeList2[18]
nodeList2[8].right_node = nodeList2[19]

nodeList2[9].left_node = nodeList2[20]
nodeList2[9].right_node = nodeList2[21]

nodeList2[1].left_node = nodeList2[4]
nodeList2[1].right_node = nodeList2[5]

nodeList2[4].left_node = nodeList2[10]
nodeList2[4].right_node = nodeList2[11]

nodeList2[10].left_node = nodeList2[22]
nodeList2[10].right_node = nodeList2[23]

nodeList2[11].left_node = nodeList2[24]
nodeList2[11].right_node = nodeList2[25]
```



```
nodeList2[5].left_node = nodeList2[12]
nodeList2[5].right_node = nodeList2[13]

nodeList2[12].left_node = nodeList2[26]
nodeList2[12].right_node = nodeList2[27]

nodeList2[13].left_node = nodeList2[28]
nodeList2[13].right_node = nodeList2[29]

nodeList2[23].left_node = nodeList2[30]
nodeList2[23].right_node = nodeList2[30]

nodeList2[24].left_node = nodeList2[31]

nodeList2[14].left_node = End
nodeList2[15].left_node = End
nodeList2[16].left_node = End
nodeList2[18].left_node = End
nodeList2[19].left_node = End
nodeList2[20].left_node = End
nodeList2[21].left_node = End
nodeList2[22].right_node = End
nodeList2[30].right_node = End
nodeList2[31].right_node = End
nodeList2[25].right_node = End
nodeList2[26].right_node = End
nodeList2[27].right_node = End
nodeList2[28].right_node = End
nodeList2[29].right_node = End

print('\nTest case 2: ')
testCase2 = Path()
print(f"Testing of searchAllPath(Start):")
testCase2.searchAllPath(Start)
print(f"Testing of searchPathByWeight(Start,End):")
testCase2.searchPathByWeight(Start,End)
```

Test case 2 output:

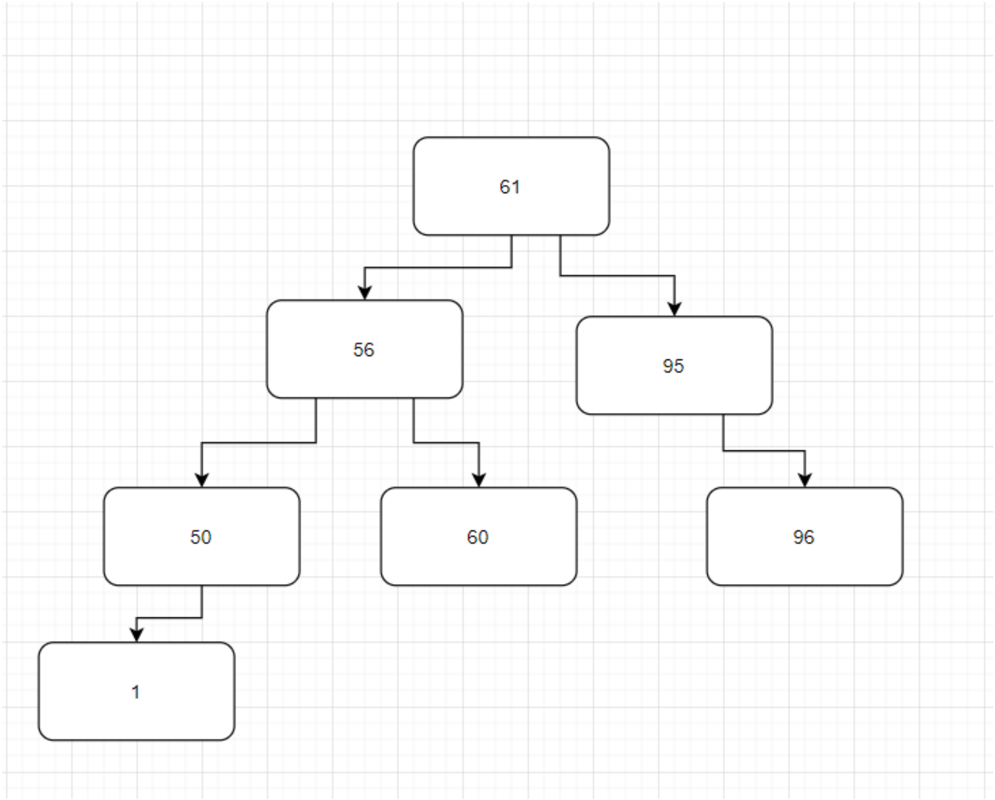


```
Test case 2:
Testing of searchAllPath(Start):
Start->1->3->7->15->End 26
Start->1->3->7->16->End 27
Start->1->3->8->17->End 29
Start->1->3->8->18->4->9->19->End       62
Start->1->3->8->18->4->9->20->End       63
Start->1->3->8->18->4->10->21->End      65
Start->1->3->8->18->4->10->22->End      66
Start->2->5->11->23->End        41
Start->2->5->11->24->31->End    73
Start->2->5->11->24->31->End    73
Start->2->5->12->25->32->End    76
Start->2->5->12->26->End        45
Start->2->6->13->27->End        48
Start->2->6->13->28->End        49
Start->2->6->14->29->End        51
Start->2->6->14->30->End        52
Testing of searchPathByWeight(Start,End):
Start->1->3->7->15->End
PS D:\hkcc_yr2\data_strucure\asm2> 
```

Answer for Question 4

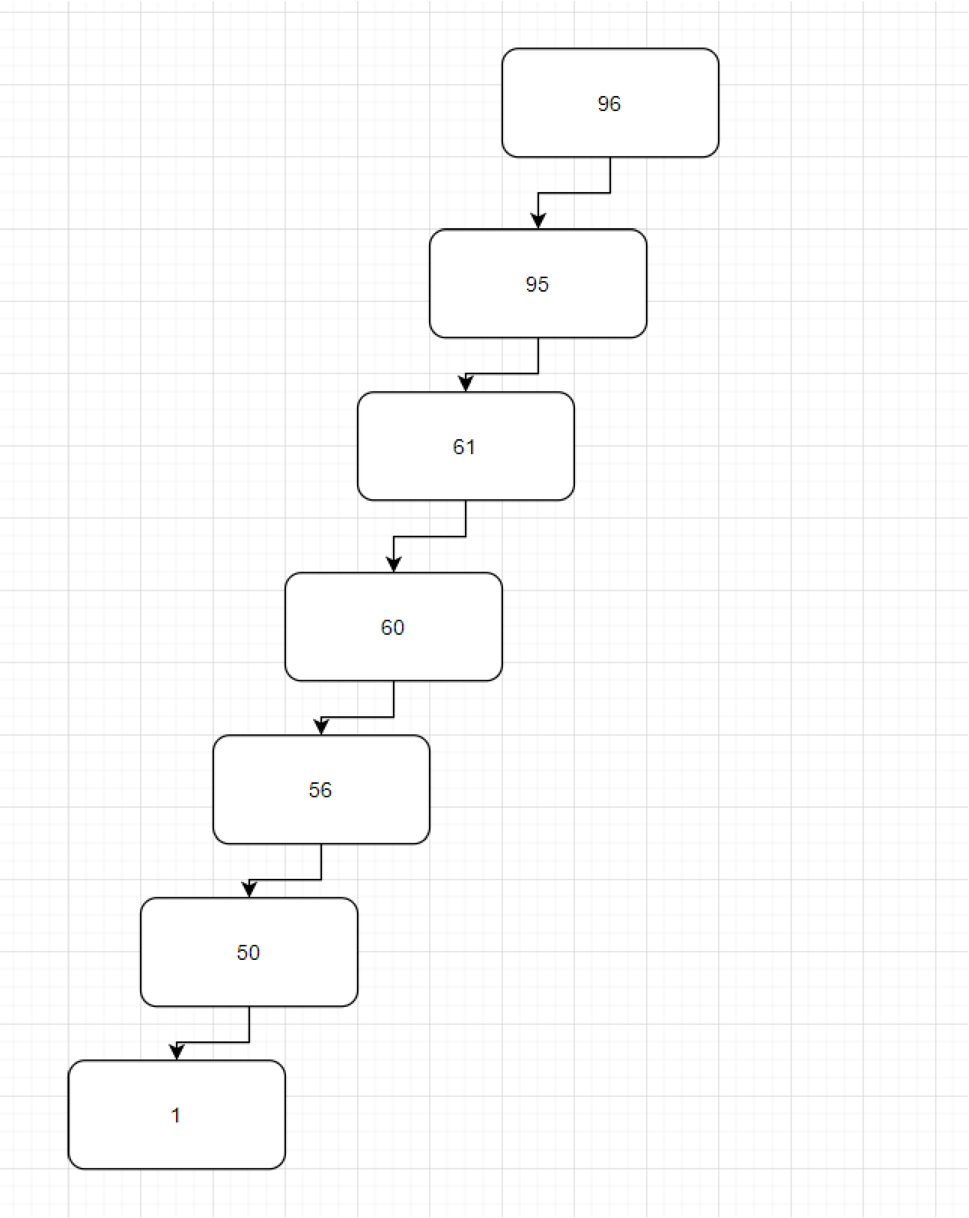
A)

a = 95
b = 56
c = 60
d = 1
e = 96
f = 50
g = 61



B)

Name:	Tsoi Yiu Chik	Student No.:	20195601A
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**18**	Total no. of pages:	**18**



C)

Insert order:

g = 61→ b = 56 → a = 95 → f = 50 → c = 60 → e = 96 →d = 1